

Text und Grafik mischen

Wer hat nicht schon einmal einen Spectrum, einen Schneider oder sogar einen Macintosh voll des Neides betrachtet, die alle scheinbar ohne Mühe Grafik und Text auf einem Screen bunt durcheinander mischen können? Wir Atari-Besitzer dagegen haben zwar einen reinen Textbildschirm (GR.0) und natürlich eine ganze Menge von verschiedenen Grafikbildschirmen, aber das Mischen von Text und Grafik endet bereits beim allseits bekannten, vierzeiligen Textfenster (von handgestrickten Display-Lists einmal abgesehen).

Wenn man die oben genannten Computer einmal genauer unter die Lupe nimmt, stellt man fest, dass die Mischbarkeit von Text und Grafik durch einen recht einfachen Trick möglich ist. Der Bildschirm ist nicht wie unser GR.0-Screen aus einzelnen Zeichen aufgebaut, sondern besteht durch und durch aus Hi-Res Grafik, mehr oder weniger dem GR.8-Screen des Atari ähnlich. Die Zeichen kommen dann durch Kopieren des entsprechenden Bitmusters per Software auf den Bildschirm. Der Vorteil dieser Lösung besteht, wie bereits gesagt, in der leichten Mischbarkeit von Text und Grafik. Außerdem wird die Hardware natürlich entsprechend einfacher, da ja nur ein Grafikmodus zu bewältigen ist. Allerdings ist dieser Vorteil teuer erkaufte, denn ein solcher Hi-Res Schirm frisst auch dann wertvollen Speicherplatz, wenn es gar nicht nötig wäre, z.B. bei der Textverarbeitung. Ganz abgesehen von den netten Spielchen, die sich mit einem echten, hardwaremäßigen Zeichensatz machen lassen. Wenn Sie z. B. schon Zeppelin oder Cavelord gesehen haben, dann wissen Sie, wovon ich rede.

Text in Hi-Res

Jetzt aber zur Sache. In der Assemblerecke beschäftigen wir uns diesmal ebenfalls mit der Mischung von Text und Grafik innerhalb eines GR.8-Schirmes. An Grafik-Befehlen mangelt es ja nicht gerade (PLOT, DRAWTO), aber mit der Ausgabe von Text ist es schlecht bestellt. Das nachfolgende Assemblerprogramm gibt Ihnen einen neuen BASIC-Befehl in die Hand, mit dem Sie einen String an (fast) jede beliebige Stelle eines GR.8-Screens drucken können. »Fast« deshalb, weil in horizontaler Richtung nur (wie in GR.0) 40 Schritte möglich sind, in vertikaler Richtung haben wir aber mit 192 Schritten die volle Auflösung zur Verfügung. Aufgerufen wird das Assemblerprogramm mit: X=USR (1536, A, L, X, Y). A bezeichnet dabei die Adresse des Strings, in BASIC einfachst durch die ADR ()-Funktion zu bekommen, L ist entsprechend die Länge des Strings, X die horizontale Position auf dem Bildschirm (0-39), Y die vertikale Position (0-183), wie üblich vom oberen Bildschirm aus gesehen. Ein praktisches Beispiel zum Aufruf des Befehls können Sie dem Demoprogramm entnehmen.

Der String wird nun nicht einfach in den Screen kopiert, sondern wird mit den Bildschirminhalt mit »Exklusiv-Oder« verknüpft. Konkret bedeutet das, dass die Buchstaben bei dunklem Hintergrund hell und bei hellem Hintergrund dunkel erscheinen. Entscheidender Vorteil dieser Methode aber ist, dass ein String ohne Zerstörung der Hintergrundgrafik vom Schirm entfernt werden kann, wenn man ihn einfach ein zweites Mal an die gleiche Stelle druckt. Auf diese Weise kommt übrigens der blinkende Text im Demoprogramm zustande. Wem dieses Feature nicht gefällt, der kann es durch Entfernen der Zeile 1190 im Assemblerlisting ausschalten.

Blick ins Innenleben

Für reine BASIC-Programmierer genügt es, das Demoprogramm ab Zeile 30000 in eigene Programme zu übernehmen. Wer Interesse an Maschinenprogrammen hat, sollte einen Blick in das wie üblich reichlich kommentierte Assemblerlisting werfen. Dort werden wie gehabt die "USR-Argumente" vom Stack genommen und anschließend die Berechnung der Bildschirmadresse aus der X- und Y-Koordinate vorgenommen. Dazu ist eine Multiplikation mit 40 notwendig, die etwas trickreich programmiert ist. Es folgt die Umrechnung des ATASCII-Zeichens in den Bildschirmcode, die Adresse des Bitmusters im Zeichensatz wird ermittelt, und schließlich wird das Bitmuster EXOR-verknüpft mit dem vorigen Bildschirminhalt ins Screen-RAM gebracht.

So, das war's. Ihr Atari kann ab jetzt auch Text und Grafik mischen. Eine recht nützliche Sache, wenn es z.B. um das Beschriften von Zeichnungen geht. Ein Tipp zum Schluss: Das Programm arbeitet auch mit veränderten Zeichensätzen, wenn deren Page-Nummer nach CHBAS (756 dez.) gepoked wurde.

Die nächste Assemblerecke in der Mai-Ausgabe wird den Newcomern unter den Assemblerprogrammierern gewidmet sein, bis dahin "Happy computing". . .

Peter Finzel

Assembler-Listing

```

0100 ;*****
0110 ;*   Text in GRAPHICS 8   *
0120 ;*                               *
0130 ;* Peter Finzel '85      *
0140 ;*****
0150 ;
0160 ;Betriebssystemadressen:
0170 ;
=0058 0180 SAVMSC = $58      Adresse des Bildschirmspeichers
=02F4 0190 CHBAS = $02F4   Pagenr. des Zeichensatzes
0200 ;

0210 ;Benutzte Speicherzellen
0220 ;
=00CB 0230 B_PTR = $CB      Zeiger in Hi-Res Bildschirm
=00CD 0240 Z_PTR = $CD      Zeiger in Zeichensatz
=00CF 0250 S_PTR = $CF      Zeiger in Textstring
=00D1 0260 B_ADR = $D1      Basisadresse des Zeichens
=00D3 0270 LAENG = $D3      Laenge des Strings
=00D4 0280 IMASK = $D4      Maske fuer Inversdarstellung
0290 ;
0300 ;*****
0310 ;Textausgaberroutine
0320 ;Aufruf: X=USR(1536,Adresse,Laenge,X,Y)
0330 ;*****
0340 ;
0000 0350  *= $0600      ist in PAGE 6, aber relocatibel
0600 68 0360 HRTXT PLA      Anz. der Args, weg damit
0601 A900 0370 LDA #0        Ein paar Register vorbesetzen
0603 85D2 0380 STA B_ADR+1    MSB Bildschirmadresse
0605 68 0390 PLA            Stringadresse in
0606 85D0 0400 STA S_PTR+1    den Textzeiger
0608 68 0410 PLA            Low-Byte
0609 85CF 0420 STA S_PTR
060B 68 0430 PLA            MSB der Laenge, in den Muell
060C 68 0440 PLA            Laenge (Max. 255 Zeichen!)
060D 85D3 0450 STA LAENG
060F 68 0460 PLA            MSB der X-Koordinate -> Muell
0610 68 0470 PLA            jetzt haben wir X
0611 85CD 0480 STA Z_PTR      aufbewahren (Z_PTR ist noch frei)
0613 68 0490 PLA            Die Y-Koord. ist ebenfalls nur
0614 68 0500 PLA            ein Byte
0615 85D1 0510 STA B_ADR      einstweilen merken
0617 0A 0520 ASL A          Multiplikation mit 40 folgt
0618 26D2 0530 ROL B_ADR+1   dazu zuerst Y*4
061A 0A 0540 ASL A
061B 26D2 0550 ROL B_ADR+1
061D 18 0560 CLC
061E 65D1 0570 ADC B_ADR      plus urspruenglichen Y-Wert
0620 85D1 0580 STA B_ADR      gibt Y*5 nach Adam Riese
0622 9002 0590 BCC M1       Uebertrag
0624 E6D2 0600 INC B_ADR+1
0610 ;
0626 0A 0620 M1 ASL A        das ganze noch mal 8
0627 26D2 0630 ROL B_ADR+1   das ist dann Y*5*8=Y*40
0629 0A 0640 ASL A
062A 26D2 0650 ROL B_ADR+1
062C 0A 0660 ASL A

062D 26D2 0670 ROL B_ADR+1
062F 18 0680 CLC
0630 65CD 0690 ADC Z_PTR      X-Koordinate addieren
0632 9002 0700 BCC M2       jetzt haben wir relative Adresse
0634 E6D2 0710 INC B_ADR+1   der Position am Schirm
0720 ;
0636 18 0730 M2 CLC
0637 6558 0740 ADC SAVMSC      Basisadresse des Bildschirm
0639 85D1 0750 STA B_ADR      muss noch dazu
063B A5D2 0760 LDA B_ADR+1
063D 6559 0770 ADC SAVMSC+1    MSB nicht vergessen...
063F 85D2 0780 STA B_ADR+1
0790 ;

```

```

0641 A5D1      0800 WEITER LDA B_ADR Bildschirmadresse
0643 B5CB      0810 STA B_PTR in Bildschirmzeiger
0645 A5D2      0820 LDA B_ADR+1
0647 B5CC      0830 STA B_PTR+1
0649 A000      0840 LDY #0
064B B4D4      0850 STY IMASK Inversmaske auf Normal
064D B4CE      0860 STY Z_PTR+1 Vorbesetzung
064F B1CF      0870 LDA (S_PTR),Y ASCII-Zeichen im Akku
                0880 ;
0651 1004      0890 BPL NORMAL Zeichen nicht invers ->
0653 A2FF      0900 LDX #$FF Inversmaske setzen
0655 B6D4      0910 STX IMASK
                0920 ;
0657 297F      0930 NORMAL AND #$7F Inversbit maskieren
0659 C960      0940 CMP #96 jetzt Umwandlung in Bildschirmcode
065B B00B      0950 BCS W_ENDE Code stimmt -->
065D C920      0960 CMP #32 ist Graphikzeichen?
065F B004      0970 BCS ALPHA nein, ist Alphazeichen -->
0661 0940      0980 ORA #64 plus 64
0663 D003      0990 BNE W_ENDE Umwandlung fertig -->
0665 38        1000 ALPHA SEC
0666 E920      1010 SBC #32 Korrektur Buchstaben und Zahlen
                1020 ;
0668 0A        1030 W_ENDE ASL A Bildschirmcode mal 8
0669 0A        1040 ASL A ist Zeiger in Zeichensatz
066A 26CE      1050 ROL Z_PTR+1
066C 0A        1060 ASL A
066D 26CE      1070 ROL Z_PTR+1
066F B5CD      1080 STA Z_PTR LSB ist fertig
0671 18        1090 CLC
0672 A5CE      1100 LDA Z_PTR+1
0674 6DF402    1110 ADC CHBAS Basisadresse des Zeichensatz
0677 B5CE      1120 STA Z_PTR+1 dazu
                1130 ;
0679 A000      1140 LDY #0 Zeichen in Hi-Res kopieren
067B A20B      1150 LDX #8 Acht Bytes kopieren
                1160 ;
067D B1CD      1170 ZEILE LDA (Z_PTR),Y eine Zeile des Zeichens
067F 45D4      1180 EOR IMASK evt. invers
0681 51CB      1190 EOR (B_PTR),Y mit vorh. Bild verknuepfen
0683 91CB      1200 STA (B_PTR),Y und eintragen
0685 18        1210 CLC
0686 A5CB      1220 LDA B_PTR naechste Zeile ist 40 Bytes
0688 692B      1230 ADC #40 Byte weiter
068A B5CB      1240 STA B_PTR
                1250 ;
068C 9002      1250 BCC M3
068E E6CC      1260 INC B_PTR+1
0690 E6CD      1270 M3 INC Z_PTR Zeichensatz Adresse (nur LSB)
0692 CA        1280 DEX
0693 D0EB      1290 BNE ZEILE Zeile fuer Zeile ...
                1300 ;
0695 E6D1      1310 INC B_ADR Bildschirmadresse fuer
0697 D002      1320 BNE M4 naechstes Zeichen vorbereiten
0699 E6D2      1330 INC B_ADR+1
069B E6CF      1340 M4 INC S_PTR Stringzeiger auf
069D D002      1350 BNE M5 naechstes Zeichen
069F E6D0      1360 INC S_PTR+1
06A1 C6D3      1370 M5 DEC LAENG ueberhaupt noch Zeichen da??
06A3 D09C      1380 BNE WEITER jawohl -->
06A5 60        1390 RTS Ciao ...

```

ASSEMBLY ERRORS: 0 23901 BYTES FREE

Basic-Programm

```

1 REM ** DEMOPROGRAMM F. HI-RES TEXT
2 REM **
10 DIM A$(100):REM Platz f. String
20 GRAPHICS 8+16:COLOR 1:REM Hi-Res ein
30 FOR I=0 TO 191:PLOT 191-I,I:DRAWTO 319-I,I:NEXT I
40 GOSUB 30000:REM * Maschinenprogramm einrichten
50 A$="Text in HI-RES Bildschirm"
60 X=6:Y=20:GOSUB 10000:REM * A$ Ausgeben
70 A$="Alle normalen Zeichen sind darstellbar"
80 X=1:Y=50:GOSUB 10000
90 A$="Drucksen Text Graphik: ♣ ♠ + ♥"
100 X=5:Y=100:GOSUB 10000
110 A$="Auch Loeschen ist moeglich..."
120 X=5:Y=150
130 GOSUB 10000:FOR I=0 TO 200:NEXT I:GOTO 130
10000 REM * GIBT A$ AN POSITION X,Y AUS
10010 Q=USR(1536,ADR(A$),LEN(A$),X,Y)
10090 RETURN
30000 REM * HIRES-TEXT Maschinenpgm
30010 S=0:RESTORE 30100
30020 FOR A=1536 TO 1701:READ D:POKE A,D:S=S+D:NEXT A
30030 IF S<>22328 THEN ? "DATEN-FEHLER!":STOP
30090 RETURN
30100 DATA 104,169,0,133,210,104,133,208,104,133,207,104,104,133,211
30110 DATA 104,104,133,205,104,104,133,209,10,38,210,10,38,210,24
30120 DATA 101,209,133,209,144,2,230,210,10,38,210,10,38,210,10,38
30130 DATA 210,24,101,205,144,2,230,210,24,101,88,133,209,165,210
30140 DATA 101,89,133,210,165,209,133,203,165,210,133,204,160,0,132
30150 DATA 212,132,206,177,207,16,4,162,255,134,212,41,127,201,96
30160 DATA 176,11,201,32,176,4,9,64,208,3,56,233,32,10,10,38,206,10
30170 DATA 38,206,133,205,24,165,206,109,244,2,133,206,160,0,162,8
30180 DATA 177,205,69,212,81,203,145,203,24,165,203,105,40,133,203
30190 DATA 144,2,230,204,230,205,202,208,232,230,209,208,2,230,210
30200 DATA 230,207,208,2,230,208,198,211,208,156,96

```

```

1 REM ** DEMOPROGRAMM F. HI-RES TEXT
2 REM **
10 DIM A$(100)
20 GRAPHICS 8+16:COLOR 1
30 FOR I=0 TO 191:PLOT 191-I,I:DRAWTO 319-I,I:NEXT I
40 GOSUB 30000
50 A$="TEXT IN HI-RES BILDSCHIRM"
60 X=6:Y=20:GOSUB 10000
70 A$="ALLE NORMALEN ZEICHEN SIND DARSTELLBAR"
80 X=1:Y=50:GOSUB 10000
90 A$="(INV I)(INV CAPS N)(INV CAPS U)(INV CAPS E)(INV CAPS
R)(INV CAPS S)(INV CAPS E)(INV CAPS R)(INV SPACE)(INV T)(INV
CAPS E)(INV CAPS X)(INV CAPS T)(INV ,) Graphik:(CTRL P) (CTRL
;) (CTRL ,) (CTRL .)"
100 X=5:Y=100:GOSUB 10000
110 A$="(INV A)(INV CAPS U)(INV CAPS C)(INV CAPS H)(INV
SPACE)(INV L)(INV CAPS O)(INV CAPS E)(INV CAPS S)(INV CAPS
C)(INV CAPS H)(INV CAPS E)(INV CAPS N)(INV SPACE)(INV CAPS
I)(INV CAPS S)(INV CAPS T)(INV SPACE)(INV CAPS M)(INV CAPS
O)(INV CAPS E)(INV CAPS G)(INV CAPS L)(INV CAPS I)(INV CAPS
C)(INV CAPS H)(INV .)(INV .)(INV .)"
120 X=5:Y=150
130 GOSUB 10000:FOR I=0 TO 200:NEXT I:GOTO 130
10000 REM * GIBT A$ AN POSITION X,Y AUS
10010 Q=USR(1536,ADR(A$),LEN(A$),X,Y)
10090 RETURN
30000 REM * MASCHINEN-UNTERPROGRAMM LADEN
30010 S=0:RESTORE 30100
30020 FOR A=1536 TO 1701:READ D:POKE A,D:S=S+D:NEXT A
30030 IF S<>22328 THEN ? "DATEN-FEHLER!":STOP
30090 RETURN
30100 DATA 104,169,0,133,210,104,133,208,104,133
30105 DATA 207,104,104,133,211
30110 DATA 104,104,133,205,104,104,133,209,10,38
30115 DATA 210,10,38,210,24
30120 DATA 101,209,133,209,144,2,230,210,10,38,210
30125 DATA 10,38,210,10,38
30130 DATA 210,24,101,205,144,2,230,210,24,101
30135 DATA 88,133,209,165,210
30140 DATA 101,89,133,210,165,209,133,203,165,210,133
30145 DATA 204,160,0,132
30150 DATA 212,132,206,177,207,16,4,162,255,134,212
30155 DATA 41,127,201,96
30160 DATA 176,11,201,32,176,4,9,64,208,3,56,233
30165 DATA 32,10,10,38,206,10
30170 DATA 38,206,133,205,24,165,206,109,244,2
30175 DATA 133,206,160,0,162,8
30180 DATA 177,205,69,212,81,203,145,203,24,165
30185 DATA 203,105,40,133,203
30190 DATA 144,2,230,204,230,205,202,208,232,230
30195 DATA 209,208,2,230,210
30200 DATA 230,207,208,2,230,208,198,211,208,156,96

```